

Научно- исследовательск ий практикум

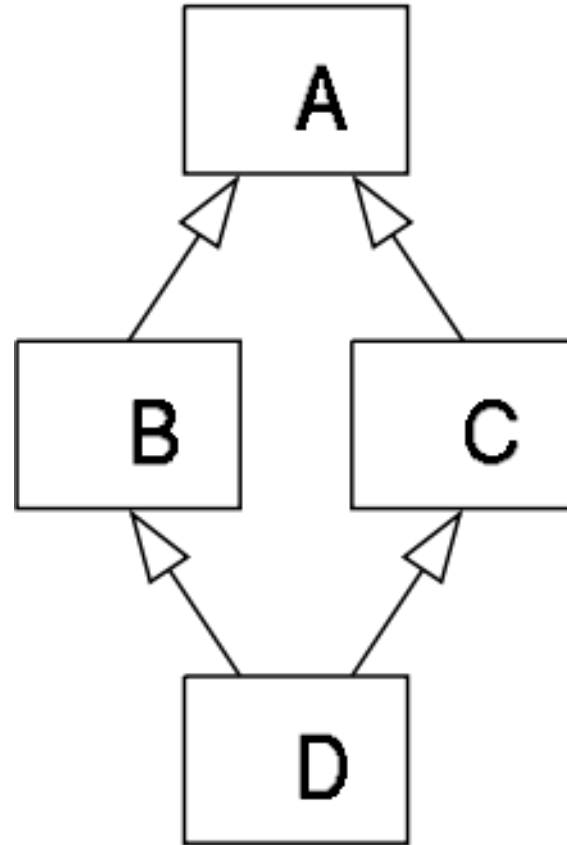
ПОЛИМОРФИЗМ. МНОЖЕСТВЕННОЕ
НАСЛЕДОВАНИЕ.

Наследование: ромб (aka diamond)

Мы уже видели примеры с множественным наследованием

И даже пример с множественным наследованием мы видели (на 8 занятии)

Как оно выглядит с виртуальными методами?



Базовый класс

```
class Base
{
public:
    Base();
    ~Base();
    void NonVirtualMethod();
    virtual void VirtualMethodA();
    virtual void VirtualMethodB();
};

Base::Base()
{
    cout << "Base default c-tor called" << endl;
}

Base::~Base()
{
    cout << "Base d-tor called" << endl;
}

void Base::NonVirtualMethod()
{
    cout << "Non-virtual method of Base class called" << endl;
}

void Base::VirtualMethodA()
{
    cout << "Virtual method A of Base class called" << endl;
}

void Base::VirtualMethodB()
{
    cout << "Virtual method B of Base class called" << endl;
}
```

Первый наследник

```
class HeirA : public Base
{
public:
    void NonVirtualMethod();
    virtual void VirtualMethodA();
};

void HeirA::NonVirtualMethod()
{
    cout << "Non-virtual method of Heir A class called" << endl;
}

void HeirA::VirtualMethodA()
{
    cout << "Virtual method A of Heir A class called" << endl;
}
```

Второй наследник

```
class HeirB : public Base
{
public:
    void NonVirtualMethod();
    virtual void VirtualMethodB();
};

void HeirB::NonVirtualMethod()
{
    cout << "Non-virtual method of Heir B class called" << endl;
}

void HeirB::VirtualMethodB()
{
    cout << "Virtual method B of Heir B class called" << endl;
}
```

main.cpp

```
int main(int argc, char *argv[])
{
    Base* ptr = nullptr;

    ptr = new Base;
    ptr->VirtualMethodA();
    ptr->VirtualMethodB();
    delete ptr;

    ptr = new HeirA;
    ptr->VirtualMethodA();
    ptr->VirtualMethodB();
    delete ptr;

    ptr = new HeirB;
    ptr->VirtualMethodA();
    ptr->VirtualMethodB();
    delete ptr;

    return 0;
}
```

Результат

```
Starting /Users/amakashov/projects/build-heir_example2.  
Virtual method A of Base class called  
Virtual method B of Base class called  
Virtual method A of Heir A class called  
Virtual method B of Base class called  
Virtual method A of Base class called  
Virtual method B of Heir B class called  
/Users/amakashov/projects/build-heir_example2-Desktop-I
```

Ещё один наследник

```
class HeirAB : public HeirA, public HeirB
{
    virtual void VirtualMethodA();
    virtual void VirtualMethodB();
    void NonVirtualMethod();
};
```


Попробуем создать указатель такой объект...

```
ptr = new HeirAB;  
ptr->VirtualMethodA();  
ptr->VirtualMethodB();  
delete ptr;
```

```
! ambiguous conversion from derived class 'HeirAB' to base class 'Base':  
  class HeirAB -> class HeirA -> class Base  
  class HeirAB -> class HeirB -> class Base  
  ptr = new HeirAB;  
      ~~~~~  
/Users/amakashov/projects/heir_example2/main.cpp  
! assigning to 'Base *' from incompatible type 'HeirAB *'  
! unused parameter 'argc' [-Wunused-parameter]  
! unused parameter 'argv' [-Wunused-parameter]
```

Почему так вышло?

Наш класс-наследник *есть (is-a)* базовый класс

В результате у нас есть базовый класс и в **HeirA** и в **HeirB**

А в **HeirAB** их оказывается целых два

Для того, чтобы избегать таких ситуаций, используется виртуальное наследование

Первый наследник

```
class HeirA : virtual public Base
{
public:
    void NonVirtualMethod();
    virtual void VirtualMethodA();
};

void HeirA::NonVirtualMethod()
{
    cout << "Non-virtual method of Heir A class called" << endl;
}

void HeirA::VirtualMethodA()
{
    cout << "Virtual method A of Heir A class called" << endl;
}
```

Второй наследник

```
class HeirB : virtual public Base
{
public:
    void NonVirtualMethod();
    virtual void VirtualMethodB();
};

void HeirB::NonVirtualMethod()
{
    cout << "Non-virtual method of Heir B class called" << endl;
}

void HeirB::VirtualMethodB()
{
    cout << "Virtual method B of Heir B class called" << endl;
}
```

Почти тот же main

```
int main(int argc, char *argv[])
{
    Base* ptr = nullptr;

    cout << "Heir A" << endl;
    ptr = new HeirA;
    ptr->VirtualMethodA();
    ptr->VirtualMethodB();
    delete ptr;

    cout << "Heir B" << endl;
    ptr = new HeirB;
    ptr->VirtualMethodA();
    ptr->VirtualMethodB();
    delete ptr;

    cout << "Heir AB" << endl;
    ptr = new HeirAB;
    ptr->VirtualMethodA();
    ptr->VirtualMethodB();
    delete ptr;

    return 0;
}
```

Результат

```
Starting /Users/amakashov/projects/build-heir_example2
Heir A
Virtual method A of Heir A class called
Virtual method B of Base class called
Heir B
Virtual method A of Base class called
Virtual method B of Heir B class called
Heir AB
Virtual method A of Heir A class called
Virtual method B of Heir B class called
/Users/amakashov/projects/build-heir_example2-Desktop-
```

Какие ещё варианты наследования бывают?

Public – объект-наследник *есть (is-a)* базовый класс

Private – объект наследник *содержит(has-a)* базовый класс

При этом все public – элементы становятся private для наследника

В этом случае доступ к унаследованным элементам возможен только внутри класса

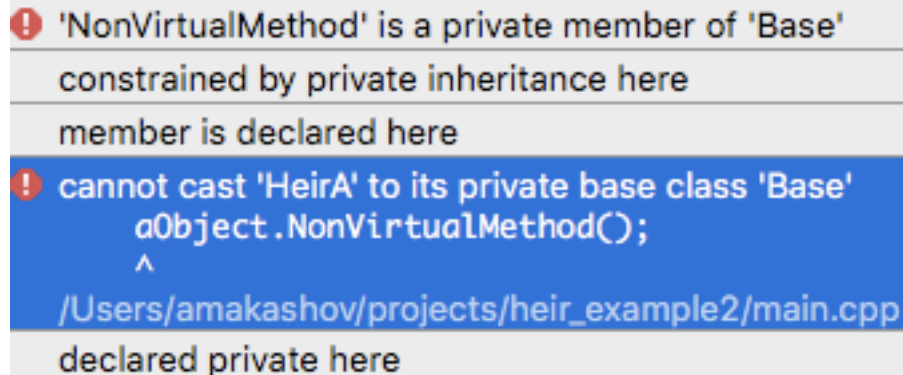
Изменим наших наследников

```
class HeirA : private Base
{
public:
    virtual void VirtualMethodA();
};

class HeirB : private Base
{
public:
    virtual void VirtualMethodB();
};
```


Попробуем вызвать метод базового класса

```
int main(int argc, char *argv[])
{
    HeirA aobject;
    aobject.NonVirtualMethod();
    return 0;
}
```



The screenshot shows two error messages from a C++ compiler. The first message is on a light gray background and states: "'NonVirtualMethod' is a private member of 'Base' constrained by private inheritance here member is declared here". The second message is on a blue background and states: "cannot cast 'HeirA' to its private base class 'Base'". Below this message, the code snippet `aobject.NonVirtualMethod();` is shown with a caret (^) pointing to the `aobject` variable. At the bottom, the file path `/Users/amakashov/projects/heir_example2/main.cpp` is displayed, followed by the text "declared private here".

```
! 'NonVirtualMethod' is a private member of 'Base'
  constrained by private inheritance here
  member is declared here
! cannot cast 'HeirA' to its private base class 'Base'
    aobject.NonVirtualMethod();
      ^
/Users/amakashov/projects/heir_example2/main.cpp
declared private here
```

Как добраться до унаследованного?

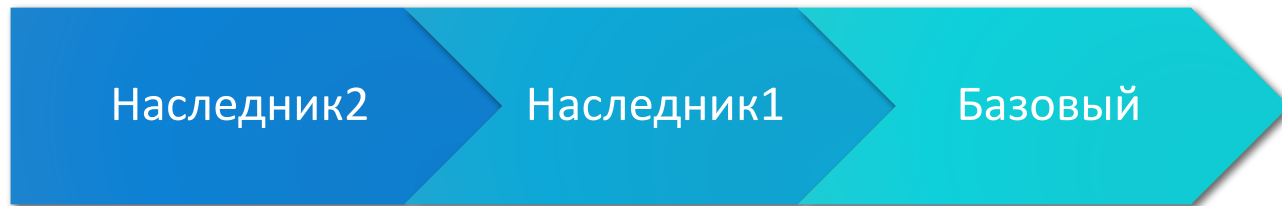
```
void HeirA::VirtualMethodA()
{
    NonVirtualMethod();
    cout << "Virtual method A of Heir A class called" << endl;
}

int main(int argc, char *argv[])
{
    HeirA aObject;
    aObject.VirtualMethodA();

    return 0;
}
```

Protected-наследование

При `privat`-наследовании следующие наследники доступ не получают:



А при `protected`-наследовании базовый класс доступен внутри Наследник2

Задание

Напишите базовый абстрактный класс Shape

Сделайте в нём виртуальные методы вычисления площади и периметра

Реализуйте наследников: классы прямоугольника, треугольника, эллипса

Реализуйте их методы вычисления площади и периметра

Сделайте возможность пользователю вводить тип фигуры и вычислять их площадь и периметр

$$L \approx 4 \frac{\pi ab + (a - b)^2}{a + b}$$